REMARKS

Claims 1, 3 and 9 to 16 are pending in the present application. Claims 1 and 9 have been amended to correct minor informalities only. Claim 16 is new. Support for claim 16 can at least be found on pages 8-11, 14-16 and 18-20 of the present application. No new matter has been added. In view of this Response, Applicant respectfully requests reconsideration of and allowance of the present application.

Interview

Applicant thanks Examiner Timblin for the courtesies extended to Applicant's representative Wesley Jones during the personal interview of January 10, 2007. A summary of the substance of the interview is set forth below.

During the interview, Applicant's representative asserted in particular that the applied §103 references did not render obvious independent claims 1, 9 and 10 and dependent claims 14 and 15. The arguments provided by Applicant's representative are substantially the same as those provided herein. No agreement as to the patentability of the claims was reached.

Claim Objections

Claim 1 is objected to because the Office Action believes step (d) which recites "when a next record requested *by* the first recordset" is inconsistent with step (a) which recites "retrieving a first record *from* a remote database memory in response to a request from a first recordset." Claim 9 is objected to for similar reasons. Applicants contend there is no need to amend step (d) to recite "from" instead of "by" as specified by the Office Action. Step (d) specifies *which* claim element requests a next record – i.e., the first recordset – while step (a) specifies *where* a first record is originally stored for retrieval – i.e., the remote database memory. Therefore, it is clear that steps (a) and (d) are directed to different aspects of the claimed invention. As such, there is no confusion created by use of the term "by" in step (d). Further, use of the term "by" in step (d) is not inconsistent with any other portion of the claim that is directed to where a records is originally stored for retrieval. Applicants therefore request that these objections be reconsidered and withdrawn.

Claim 9 has been amended to recite "*steps* (a) and (b)" as requested on page 2 of the Office Action. Applicants therefore request that this objection be reconsidered and withdrawn.

Claim 10 is objected to for using a semicolon at the end of the second determining step. Applicants contend that the semicolon distinguishes the two alternative steps that depend on

the results of the determining step. Thus, the semicolon performs the same function as the semicolon used at the end of the first determining step in the claim. The use of the first semicolon was not objected to in the Office Action. Applicants therefore request that this objection be reconsidered and withdrawn as use of the semicolon is consistent with a use that is not objected to by the Office Action. If this objection is to be maintained, then Applicant requests the Examiner to specify the need to remove the semicolon with more particularity.

35 U.S.C. § 112 Rejections

Claims 1 and 9 have been amended as requested on pages 2 and 3 of the Office Action. Applicant therefore requests that the rejections of claims 1 and 9 under § 112 be reconsidered and withdrawn.

Prior Art Rejections

Claims 1, 3, 9, 11 and 12 stand rejected under 35 U.S.C. § 103(a) as being unpatentable over *Nguyen et al.* (U.S. Patent No. 6,216,137) in view of *Kodavalla et al.* (U.S. Patent No. 5,717,919). Claims 10 and 13-15 stand rejected under 35 U.S.C. § 103(a) as being unpatentable over *Kodavalla et al.* in view of *Nguyen et al.* Applicant respectfully requests withdrawal of these rejections because neither *Kodavalla et al.* nor *Nguyen et al.*, either alone or in combination, teaches or suggests all elements of independent claims 1, 9 and 10.

Neither Kodavalla et al. nor Nguyen et al teach or disclose the management of bufferpages and redundant copies of records in a local memory associated with a mobile device application as claimed

Independent claims 1, 9 and 10 each include features directed to a mobile client application retrieving records from a remote database and subsequently storing the retrieved records in a local buffer memory associated with the mobile client application. Specifically, claim 1 recites, *inter alia*:

> A method for managing bufferpages and redundant copies of records in a local memory associated with a mobile device application, comprising:
>
> (a) retrieving a first record **from a remote database memory** in response to a request from a first recordset;
>
> (b) saving the first record on a first bufferpage of **the local memory associated with the mobile device application**, the

first bufferpage being associated with the first recordset;

Claim 10 recites, *inter alia*:

> A system for managing bufferpages and redundant copies of records of a mobile device application, comprising:
> **a remote database memory**;
> **a local program memory associated with the mobile device application**; and
> a local mobile processor coupled to the remote database memory and the local program memory associated with the mobile device application, the local mobile processor adapted to:
> (a) retrieve a first record **from the remote database memory** in response to a request from a first recordset;
> (b) save the first record on a first bufferpage of **the local program memory associated with the mobile device application**, the first bufferpage being associated with the first recordset;

Claim 10 recites, *inter alia*:

> A method of managing **fixed units of buffer memory associated with a mobile client application**, comprising:
> retrieving a record stored in **a remote database memory**;
> . . .
> saving the retrieved record in **the current fixed unit of buffer memory** if the size of the retrieved record is smaller than the freespace of the current fixed unit of buffer memory;

Neither *Kodavalla et al.* nor *Nguyen et al.*, either alone or in combination, teach this aspect of independent claims 1, 9 and 10 as neither reference is directed to a mobile client application locally storing records retrieved from a remote database.

With respect to claim 1, the Office Action issued May 8, 2007 admitted on page 4 that "Kodavalla . . . does not explicitly teach a mobile device application and local memory associated with the mobile device application." With respect to claim 9, the Office Action issued May 8, 2007 admitted on page 6 that "Kodavalla fails to expressly teach a local program memory associated with the mobile device application and a local mobile processor coupled to the remote database memory and the local program memory associated with the mobile device application." Accordingly, the present Office Action alleges that newly cited reference *Nguyen et al.* teaches these features.

To support these allegations, however, the Office Action has clearly mischaracterized the *Nguyen et al.* reference. This is evident from the citations to *Nguyen et al.* provided in the Office Action. For example, the Office Action on pages 3-4 alleges that the disclosure of a stylus and touch screen by *Nguyen et al.* at col. 4, lines 33-39 "suggest[s] a mobile device." By reviewing this disclosure in its fuller and proper context, the mischaracterization by the Office Action is clear. From col. 3, line 66 to col. 4, line 57, *Nguyen et al.* describes a "computer system upon which the preferred embodiment of the present invention can be implemented" (col. 3, line 66-col. 4, line 1). Throughout col. 4 of *Nguyen et al.*, a computer architecture including input devices and output devices are described. At no point in the description of the computer system does *Nguyen et al.* suggest that the computer system is a mobile device. As a result, the Office Action attempts to use a citation taken out of context to reach the tenuous conclusion that a mobile device is contemplated. As a stylus and touch screen can be used with an immobile computing device, and do not by themselves necessitate a mobile device, and further because *Nguyen et al.* fails to explicitly disclose or suggest mobile devices, the Office Action's allegation that these features suggest a mobile device is clearly unfounded and unsupported by *Nguyen et al.*

Further, the Office Action fails to indicate any portion of *Nguyen et al.* that discloses or suggests the retrieval of records from a remote database. The Office Action alleges that Figure 1b of *Nguyen et al.* discloses the retrieval of records from a remote database. Again, the Office Action's citations to *Nguyen et al.* are mischaracterized to enable tenuous conclusions to be made on the scope of *Nguyen et al.'s* disclosure. No portion of Figure 1b suggests a remote database. Element 188 is labeled "DATA" but fails to disclose where the data is stored, or more specifically, that the data is stored in a database remote from a mobile application that is accessing the data. Further, the corresponding description of Figure 1b by *Nguyen et al.* at col. 5, lines 3-26 fails to disclose or suggest the retrieval of records from a remote database. In fact, the terms "remote" and database" never appear in relation to the description of Figure 1b. Therefore, the allegation that Figure 1b discloses the retrieval of records from a remote database by a mobile client application is entirely unsupported.

With respect to claim 10, Applicants contend it is clear that the feature of storing a record retrieved from a remote database in buffer associated with a mobile client application is recited. Though the prior Office Action clearly admitted that this feature is not disclosed by *Kodavalla et al.* as recited in claims 1 and 9, the current Office Action alleges that *Kodvalla et al.* does disclose the recitation of this feature in claim 10. However, no portion of *Kodavalla et al.*

teaches or suggests the storage of records in a buffer memory associated with a mobile client application as recited in claim 10. As reproduced above, claim 10 recites "saving the retrieved record in the current fixed unit of buffer memory." From the preamble of claim 10, it is clear that the fixed unit of buffer memory is "associated with a mobile client application." Since the body of claim 10 relies on the preamble for completeness, the preamble is afforded patentable weight as it specifies the context for where retrieved records are stored. Therefore, claim 10 recites the storage of retrieved records from a remote database in a local buffer associated with a mobile client application which is a feature that the prior Office Action has clearly admitted is not disclosed by *Kodavalla et al.*

For at least these reasons, neither *Kodavalla et al.* nor *Nguyen et al.* teaches or discloses retrieving records from a remote database and subsequently storing the retrieved records in a local buffer memory associated with the mobile client application as recited in independent claims 1, 9 and 10.

Neither Kodavalla et al. nor Nguyen et al teach or disclose the storing of pointers between redundant copies of records as claimed

Independent claims 1, 9 and 10 each include features directed to managing redundant copies of database records. Specifically, claim 1 is a method claim that recites, *inter alia*:

> (e) **determining if one of the first record, the at least one further record, and the next record was previously retrieved and saved** in the local memory associated with the mobile device application by at least one of the first recordset and at least one second recordset **as one of a prior saved version of the first record, a prior saved version of the at least one further record, and a prior saved version of the next record, respectively**; and
>
> (f) **storing a pointer with one of the prior saved version of the first record, the prior saved version of the at least one further record, and the prior saved version of the next record,** the pointer pointing to the one of the first record, the at least one further record, and the next record if one of the first record, the at least one further record, and the next record was previously retrieved and saved as one of the prior saved version of the first record, the prior saved version of the at least one further record, and the prior saved version of the next record, respectively, otherwise creating a business object kernel pointing to one of the first record, the at least one further record, and the next record.

Claim 9 is a system claim that recites similar features as those highlighted above for claim 1.

Claim 10 is a method claim that recites, *inter alia:*

> **determining if the retrieved record was previously retrieved and stored** by the mobile client application and:
>
> **storing a pointer pointing from a fixed unit of buffer memory storing a most recent copy of the retrieved record to a fixed unit of buffer memory storing a new copy of the retrieved record**, if the retrieved record was previously retrieved and stored by the mobile client application;

Neither *Kodavalla et al.* nor *Nguyen et al.*, either alone or in combination, teach the mechanism of managing redundant records as recited in claims 1, 9 and 10.

Contrary to the allegations of the Office Action, *Nguyen et al.* fails to teach the storing of pointers between redundant copies of records (i.e., prior saved versions of records or prior saved copies of records) as recited in independent claims 1, 9 and 10. The claimed term "records" is clearly defined in the specification and understood to mean a row of a database table that comprises data organized into fields (*See* pg. 2, lines 1-5 and pg. 14, lines 30-34 of the present application). *Nguyen et al.* is in no way directed to the management of redundant copies of such database records. Instead, *Nguyen et al.* is directed to a mechanism by which data formatted according to a first data type version can be converted to the formatting of a second data type version. This allows a newer version of a software program the ability to access data stored according to a format from an older version of the software program.

Nguyen et al., at col. 1, lines 20-39, explains the changes a data type may experience as a software package changes over time:

> As a software package evolves, numerous versions may be created for the same data type. **For example, a first version of a software package may be designed to operate on data that is formatted according to a first version of a data type, while a second version of the same software package is designed to operate on data that is formatted according to a second version of the data type.** All of the versions of a particular data type are referred to as a "schema". A particular version of a data type is referred to as a "schema version". The process of moving from one version of a schema to another version of the schema is referred to as schema evolution. **The format of a data type may be modified in a variety of ways during the schema evolution process. For example, new attributes may be added to a data type, existing attributes may be removed from a data type, and the type of data contained in particular attributes may be changed.** The structure (e.g. the set of attributes and type of attributes) of a schema version is referred to as the "format" of the schema version. (Emphasis

added)

It is clear from the above description that *Nguyen et al.'s* use of the term "version" refers to the type of data formatting associated with a particular software version and, in contrast to the present invention, does not refer to multiple copies of the same data stored according to the same formatting. In order to be able to convert data from a first version type to a second data type, *Nguyen et al.*, at col. 5, line 49 to col. 6, line 44 describes a process by which a software application "register[s] the versions of the data types (the 'types') used by its libraries when the application is initialized." (col. 5, lines 50-52) During registration, a table is generated that stores data formatting information for each data type version used by the application. This data formatting information is not a database record of data organized into fields but is instead control information used to specify how raw data is to be organized and formatted – i.e., it is not raw data values from a database record itself (*See* col. 6, lines 48-50 which contrasts the information of the registration table to "instances" which are raw data stored according to a particular data type version). Further, it is clear from col. 6, lines 10-18 of *Nguyen et al.* that redundant copies of formatting information are not possible:

> When inserting an entry for a data type into the type version table 212, each registration routine checks the type version table 212 to determine if the type version table 212 already contains an entry for the data type. If an entry for the data type already exists, then the application 200 determines whether the version to be registered matches the version that is already registered. **No new entry is added if the versions match, since the new entry would simply duplicate the existing entry**. If the versions do not match, then a type conflict exists and execution of the application is halted. (Emphasis added)

From the above, it is clear that *Nguyen et al.* is in no way directed to the management of redundant copies of database records that are stored according to the same format. Further, it is clear that the different versions of information stored by *Nguyen et al.* contain control or format information exclusively and is not raw data values organized into fields. While formatting information for different data type versions can be linked together, *Nguyen et al.* fails to disclose "instances" of stored data, formatted according to the same formatting version, being linked together. Lastly, it is clear that *Nguyen et al.* explicitly specifies a mechanism by which redundant copies of control information cannot be stored.

The Office Action notes on pages 14 and 15 that Applicant has stated that the claimed

invention specifies the use of pointers pointing to *individual* records but has not recited the term "individual" in the claims. Applicants contend that records is clearly defined in the specification of the present application to be a row of a database table. In view of the specification and the plain language of the claims, it is evident that a record is a singular or individual portion of data. Therefore, use of the term "individual" to modify record is redundant and unnecessary.

For at least these reasons, *Nguyen et al.* fails to teach or disclose the storing of pointers between redundant copies of records (i.e., prior saved versions of records or prior saved copies of records) as recited in independent claims 1, 9 and 10.

Neither Kodavalla et al. nor Nguyen et al teach or disclose the creation of business object kernels pointing the first instance of a saved record as claimed

The Office Action admits on page 5 that *Nguyen et al.* fails to teach or disclose "creating a business object kernel pointing to one of the first record, the at least one further record, and the next record" when one of the first record, the at least one further record, and the next record has not been previously retrieved and stored as recited in claim 1. Claims 9 and 10 recite similar limitations. Contrary to the Office Action's allegations, *Kodavalla et al.* fails to teach or suggest this feature of the claimed invention.

Kodavalla et al. does not specify the use of pointers to point to individual records. The only use of pointers disclosed by *Kodavalla et al.* is their use to link page chains which hold multiple records:

> As shown in FIG. 3A, the data records or rows of a database table are actually stored in a particular structure known as a data page. **A data page may be viewed as a storage unit (*e.g.*, 2K storage block) which holds one or more records, such as page 310.** When a data page is "full," typically on the order of about 50 to 100 records, it is necessary to allocate a new data page. **Every page which is allocated is linked to its previous and next neighboring pages via forward and backward page pointers (*e.g.*, pointers 311, 321), so that logically a chain (linked list) of (physical) pages exists.** This forms the "page chain," such as the page chain 300 shown in FIG. 3A. Typically, identifiers or "Page IDs" for the first and last page of a page chain are maintained in a system catalog. (Col. 7, lines 29-42)

As described above, pointers are not disclosed by *Kodavalla et al.* as being used to point to a specific record. Instead, they are clearly only used to point to a "data page." Further, as

Kodavalla et al. fails to teach or disclose the use of pointers to point to individual records, *Kodavalla et al.* necessarily cannot teach or disclose the creation of a business object kernel to point to a record when it is determined that the record has not been previously retrieved and saved, as recited in claims 1, 9 and 10.

The Office Action cites to several sections of *Kodavalla et al.* that are in no way related to the creation of a business object kernel pointing to the first instance of a saved record as recited in claim 1, 9 and 10. Specifically, the Office Action cites to col. 4, line 35 of *Kodavalla et al.* which recites "an input/output controller 103, a keyboard 104 and a pointing device 105 (e.g., mouse, track ball, pen device, or the like)." Clearly, this section of *Kodavalla et al.* is describing a computer system that may embody the invention of *Kodavalla et al.* and in no way describes the creation of business kernels pointing to first instances of saved records. The Office Action also refers to col. 1 line 60 which recites "each day more and more businesses are run from mission-critical systems which store information on server-based SQL database systems, such as Sybase SQL Server." It is entirely unclear as to why this section of *Kodavalla et al.* is cited as it in no way can be construed as disclosing the creation of business kernels pointing to first instances of saved records. Lastly, the Office Action cites to col. 20, line 55 which provides software code for allocating a page chain. As mentioned above, a page chain is not a record. Further, no portion of the software code explicitly teaches or discloses the creation of business kernels pointing to first instances of saved records as recited in claims 1, 9 and 10.

For at least these reasons, *Kodavalla et al.* fails to teach or disclose the creation of business object kernels pointing the first instance of a saved record as recited in independent claims 1, 9 and 10.

For at least the reasons provided above, Applicants assert that claims 1, 9 and 10 are allowable over *Kodavalla et al.* and *Nguyen et al.*

Claims 3 and 11 depend from independent claim 1 and are allowable for at least the reasons applicable to claim 1, as well as due to the features recited therein.

Claim 12 depends from claim 9 and claims 13 to 15 depend from claim 10 and are allowable for at least the reasons applicable to claims from which they respectively depend, as well as due to the features recited therein.

Neither Kodavalla et al. nor Nguyen et al teach or disclose the subject matter of dependent claims 14 and 15

Applicant notes in particular that *Kodovalla et al.* clearly fails to teach or suggest the features of dependent claim 14 as alleged in the Office Action. Regarding claim 14, the Office Action has failed to show how *Kodovalla et al.* teaches or suggests storing a created business object kernel in a look-up table. Each of the cites to *Kodovalla et al.* provided by the Office Action refer to the use of a system catalog to access a data page comprising multiple individual records. However, none of the cites, nor any other portion of *Kodavalla et al.*, explicitly specifies that a system catalog stores a business object kernel. Further, none of the cites, nor any other portion of *Kodavalla et al.*, explicitly specifies that a system catalog or other table stores a business object kernel pointing to a specific, individual record and not simply a data page of records.

Applicant additionally notes that *Nguyen et al.* clearly fails to teach or suggest the features of dependent claim 15 as alleged in the Office Action. Claim 15 specifies the business object kernel including a counter that indicates the number of times the retrieved record has been (redundantly) stored. It is clear that the counter, as claimed and understood in the relevant art, comprises a numerical value that is increased each time a particular record is retrieved and re-saved. The Office Action alleges that the version number identifiers of the schema records depicted in Figure 4a teach or disclose the counter recited in claim 15. Applicants contend that in no way can the collective storage of different version identifiers for multiple schema version records be reasonably construed to be a counter as that terms is claimed and understood in the art. That is, the version number label for each record does not function as a counter – the version label only specifies a particular software version. Further, the version number label does not indicate (e.g., by itself being incremented) how many times a specific record from a database table has been previously retrieved and stored – i.e., re-saved – thereby creating a redundant record. Therefore, the use of version labels does not teach or disclose the counter as recited in claim 15.

Request for Interview

In order to advance prosecution, Applicant hereby requests an interview be conducted between Applicant's representative Wesley Jones, the Examiner and the Examiner's supervisor if the Examiner does not believe one or more claims of the present application are in condition for allowance in view of the arguments presented in this paper. Applicant respectfully requests the Examiner to contact Applicant's undersigned representative at the number provided below to arrange the interview based on the Examiner's availability and the availability of the

Examiner's supervisor.

CONCLUSION

Applicant respectfully requests entry of the above amendments and favorable action in connection with this application. The Office is hereby authorized to charge any additional fees or credit any overpayments under 37 C.F.R. 1.16 or 1.17 to Kenyon & Kenyon Deposit Account No. 11-0600. The Examiner is invited to contact the undersigned at (202) 220-4419 to discuss any matter concerning this application.

Applicant respectfully submits that all claims are allowable and requests allowance of the present application.

Respectfully submitted,

KENYON & KENYON LLP

Dated: January 16, 2008

By: /Wesley W. Jones/
Wesley W. Jones
(Reg. No. 56,552)

KENYON & KENYON LLP
1500 K Street, NW, Suite 700
Washington, DC 20005-1257
Telephone: (202) 220-4419
Facsimile: (202) 220-2201
701674_1.DOC